UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/754,785	01/04/2001	Pierre-Alain Darlet	11283/35	3238

30636 7590 08/03/2007

FAY KAPLUN & MARCIN, LLP
150 BROADWAY, SUITE 702
NEW YORK, NY 10038

EXAMINER	
KISS, ERIC B	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
08/03/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	09/754,785	DARLET, PIERRE-ALAIN	
	Examiner	Art Unit	
	Eric B. Kiss	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *23 July 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-60* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-60* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114 was filed in this application after a decision by the Board of Patent Appeals and Interferences, but before the filing of a Notice of Appeal to the Court of Appeals for the Federal Circuit or the commencement of a civil action. Since this application is eligible for continued examination under 37 CFR 1.114 and the fee set forth in 37 CFR 1.17(e) has been timely paid, the appeal has been withdrawn pursuant to 37 CFR 1.114 and prosecution in this application has been reopened pursuant to 37 CFR 1.114. Applicant's submission filed on July 23, 2007, has been entered.

Claims 1-60 are pending.

Response to Arguments

2. Applicant's arguments with respect to claims 1-60 have been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

4. Claims 1-15 and 40-60 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Amended claims 1, 9, and 55 contain the limitation, "displaying an output of the software module to a user" (Claims 1 and 55,) and "a display to display at least an output of the software module to a user," (Claim 9.) This limitation does not appear to have clear support, either expressly or inherently, in the original disclosure (including the originally filed claims and drawings). The only apparent mention of displaying given in the original disclosure appears to be the displaying/printing of error messages associated with a linking process, (Specification at p. 19, paragraph 3; *Id.* at p. 26, paragraph 2,) which is not the same as displaying any actual output of the software module itself. Accordingly, these limitations are considered new matter.

Claim Rejections - 35 USC § 101

5. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claims 1-15, 40, 41, and 43-56 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Rather than repeat the extensive rejection under 35 U.S.C. § 101 set forth by the Board of Patent Appeals and Interferences, reference is made to the Decision on Appeal mailed May 23, 2007. (Decision on Appeal (05/23/2007) at pp. 12-31.) Because applicant's amendments to independent claims 1, 9, and 55 do not find clear support in the original disclosure (see the rejection of claims 1-15 and 40-60, *supra*), it is not clear whether the claims as amended require machine implementation, nor is it clear that any physical transformation would result from practicing the claimed invention. Accordingly, the rejection set forth in the Decision on Appeal has not been appropriately addressed by applicant's amendments, and the rejection is maintained.

7. To expedite a complete examination of the instant application, the claims rejected under 35 U.S.C. §101 (non-statutory) above are further rejected as set forth below in anticipation of Applicant amending these claims to place them within the four statutory categories of invention.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

9. Claims 16-39 are rejected under 35 U.S.C. 102(b) as being anticipated by John Levine, "Linkers and Loaders, chapter 6," June 1999 [online] accessed 08/15/2005, Retrieved from Internet <URL: http://www.iecc.com/linker/linker06.txt>, 9 pages (hereinafter *Levine*).

As per claim 16, *Levine* discloses receiving a software module sequentially, the software module having at least one symbol reference; locating a section header table of the software module (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)); linking the software module onto a target memory space; and resolving the at least one symbol reference, using the section header table (i.e., using the archive header and file headers) without storing the entire software module in local memory while the symbol reference is resolved (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references; see also pp. 2 and 4 (describing a computer structure having a "64-bit architecture" that runs a version of the UNIX® operating system); Decision on Appeal (05/23/2007) at p. 10).

As per claims 17-22, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claim 23, *Levine* discloses a linker configured to sequentially receive a software module having at least one symbol reference, the linker configured to locate a section header table of the software module (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)), the linker configured to resolve the symbol reference using the section header table (i.e., using the archive header and file headers), the linker configured to store less than the entire software module in local memory during the resolution of the at least one symbol reference (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 24-27, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5; see further, "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 28 and 29, *Levine* further discloses a system symbol table including a field indicative of a defining software module (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claims 30 and 31, *Levine* further discloses a software module list (see "Library formats" on pp. 1-5).

As per claims 32-34, *Levine* further discloses storing link status in a symbol table (see "Library formats" on pp. 1-5).

As per claim 35, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claim 36, *Levine* discloses receiving a software module sequentially, the software module having at least one symbol reference; locating a section header table of the software module (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)); linking the software module onto a target memory space; and resolving the at least one symbol reference, using the section header table (i.e., using the archive header and file headers) without storing the entire software module in local memory at one time (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, the use of a computer-readable medium (such as memory) is inherent (and necessary) in realizing the computer-implemented functionality described in *Levine*.

As per claim 37, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claim 38, *Levine* disclose receiving a software module, the software module including references to locations within the software module, at least some of the references being backward references; and reordering the components of the software module into a

predetermined order to remove at least some of the backward references (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references), wherein the components include at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)). Further, the use of a computer-readable medium (such as memory) is inherent (and necessary) in realizing the computer-implemented functionality described in *Levine*.

As per claim 39, *Levine* discloses receiving a software module sequentially, the software module having at least one symbol reference; locating a section header table of the software module (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)); linking the software module onto a target memory space; and resolving the at least one symbol reference, using the section header table (i.e., using the archive header and file headers) without storing the entire software module in local memory at one time (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, the use of a computer-readable medium (such as memory) is inherent (and necessary) in realizing the computer-implemented functionality described in *Levine*.

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 1-15, 40, 41, and 43-60 are rejected under 35 U.S.C. 103(a) as being unpatentable over John Levine, "Linkers and Loaders, chapter 6," June 1999 [online] accessed 08/15/2005, Retrieved from Internet <URL: http://www.iecc.com/linker/linker06.txt>, 9 pages (hereinafter *Levine*) in view of "UNIX man pages : ar ()," 1991, Free Software Foundation, Inc., [online] accessed 08/1/2007, Retrieved from Internet <URL: http://www.eskimo.com/cgi-bin/man-cgi?ar>, 5 pages (hereinafter *man_ar*).

As per claim 1, *Levine* discloses receiving a software module, the software module including references to locations within the software module, at least some of the references being backward references; and reordering components of the software module into a predetermined order to remove at least some of the backward references (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references), wherein the components include at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)). Although *Levine* fails to expressly disclose displaying an output of the software

module to the user, *man_ar* teaches that the *ar* command disclosed in *Levine* has known command-line options to readily generate such an output (see, e.g., the description of the "p" (Print) operation on p. 2). Merely making use of one of such a known and well-documented feature for its intended purpose would have been obvious to one of ordinary skill in the art at the time the invention was made.

As per claim 2, *Levine* further discloses adjusting at least one of the references in the software module to reflect the reordering of the components of the software module, so that the at least one of the references remains a reference to the same component, by to the component's new, reordered location, the new, reordered location coming after the at least one reference in the software module (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claims 3 and 4, *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claims 5-8, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5). *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library

in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claim 9, *Levine* discloses a reorder module configured to receive a software module including references to locations within the software module, at least some of the references being backward references, the reorder module configured to reorder components of the software module into a predetermined order and remove at least some of the backward references (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references), the components including at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)). Although *Levine* fails to expressly disclose displaying an output of the software module to the user, *man_ar* teaches that the *ar* command disclosed in *Levine* has known command-line options to readily generate such an output (see, e.g., the description of the "p" (Print) operation on p. 2). Merely making use of one of such a known and well-documented feature for its intended purpose would have been obvious to one of ordinary skill in the art at the time the invention was made.

As per claims 10, *Levine* further discloses adjusting at least one of the references in the software module to reflect the reordering of the components of the software module, so that the

at least one of the references remains a reference to the same component, by to the component's new, reordered location, the new, reordered location coming after the at least one reference in the software module (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claims 11 and 12, *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claims 13-15, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5). *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claims 40 and 41, *Levine* further discloses linking the reordered module after the reordering (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and

lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 43-46, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claims 47-54, *Levine* further discloses the reference pointing to/into a section or module before and after reordering (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claim 55, *Levine* discloses receiving a software module, the software module including components arranged in a first order, a first one of the components including a reference to a location in a second one of the components, the second one of the components preceding the first one of the components in the first order; and arranging the components into a predetermined second order so that the second one of the components is subsequent to the first one of the components in the second order (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references), wherein the components include at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)). Although *Levine* fails to expressly disclose displaying an output of the software module to the user, *man_ar* teaches that the *ar* command disclosed in *Levine* has known command-line options to readily generate such an output (see, e.g., the description of the "p" (Print) operation on p. 2).

Merely making use of one of such a known and well-documented feature for its intended purpose would have been obvious to one of ordinary skill in the art at the time the invention was made.

As per claims 56 and 57, *Levine* further discloses linking the reordered module after the reordering (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claim 58-60, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

12. Claim 42 is rejected under 35 U.S.C. 103(a) as being unpatentable over *Levine* and *man_ar*, as applied to claim 1 above, and further in view of U.S. Patent No. 6,185,733 to Breslau et al.

As per claim 42, *Levine* discloses such a method but fails to expressly disclose transferring the reordered module to a different computer system and linking the module on the different computer system. However, *Breslau et al.* teaches the use of remote object libraries distributed prior to linking (see, for example, col. 4, lines 11-20). Therefore, it would have been obvious to one of ordinary skill in the computer art at the time the invention was made to such use of a different computer for linking. One would be motivated to do so, for example, to facilitate distributed software development efforts or reduce the physical storage requirements for object files (see, for example, col. 2, lines 4-25).

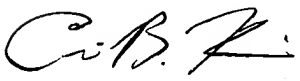
Conclusion

Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Eric B. Kiss whose telephone number is (571) 272-3699. The Examiner can normally be reached on Tue. - Fri., 7:00 am - 4:30 pm. The Examiner can also be reached on alternate Mondays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Tuan Dam, can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is (571) 273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Any inquiry of a general nature should be directed to the TC 2100 Group receptionist: 571-272-2100.



Eric B. Kiss
August 1, 2007